An Algorithm for the Computation of Linear Forms

J. E. Savage[1]
Communications Systems Research Section

Many problems, including matrix-vector multiplication and polynomial evaluation, involve the computation of linear forms. An algorithm is presented here that offers a substantial improvement on the conventional algorithm for this problem when the coefficient set is small. In particular, this implies that every polynomial of degree n with at most s distinct coefficients can be realized with $O(n/\log_s n)$ operations. It is demonstrated that the algorithm is sharp for some problems.

I. Introduction

How many operations are required to multiply a vector by a known matrix or evaluate a known polynomial at one point? Such questions are frequently asked and Winograd (Ref. 1) has shown the existence of real matrices and polynomials (containing indeterminates over the rationals, for example) for which the standard matrix-vector multiplication algorithm and Horner's rule for polynomial evaluation are optimal. That is, n^2 real multiplications and $n(n-1)$ additions are required for some $n \times n$ matrices to multiply the matrix by an n-vector, and n multiplications and n additions are required by some polynomials of degree n to evaluate the polynomial. In this paper, we use an algorithm for the computation of "linear forms" to show that the $n \times n$ matrix-vector multiplication problem and the polynomial evaluation problem can be solved with $O(n^2/\log_s(n))$ and $O(n/\log_s(n))$ operations, respectively, when the matrix entries and the polynomial coefficients are known and drawn from a set of size s (even when the entries and coefficients are variables). These results are obtained by exhibiting potentially different algorithms for each matrix and each polynomial.

The algorithm presented here for the computation of "linear forms" is very general and can be applied to many problems including matrix-matrix multiplication, the computation of sets of Boolean minterms, of sets of product over a group, as well as the two problems mentioned above. Applications of this sort are discussed in Section III.

We now define "linear forms." Let S and T be sets and let R be a "small" finite set of cardinality $|R| = s$. Let $\cdot : R \times S \rightarrow T$ be any map (call it multiplication) and let $+ : T \times T \rightarrow T$ be any associative binary operation (call

[1]Consultant from Brown University; Division of Engineering.

it addition). Then the problem to be considered is the computation of the m "linear forms" in $x_1, x_2, \cdots, x_n$

$$a_{i1} \cdot x_1 + a_{i2} \cdot x_2 + \cdots + a_{in} \cdot x_n, \qquad 1 \leq i \leq m$$

where $a_{ij} \varepsilon R$ and $x_j \varepsilon S$. The elements in R shall be regarded as symbols that may be given any interpretation later. For example, in one interpretation, R may be a finite subset of the reals and, in another, R may consist of s distinct variables over a set Q, say.

An algorithm is given in the next section that for each $m \times n$ matrix of coefficients $A = \{a_{ij}\}$ evaluates the set

$$L_A(\mathbf{x}) = \left\{ \sum_{j=1}^{n} a_{ij}x_j \,|\, 1 \leq i \leq m \right\}$$

of linear forms with $O(mn/\log_s(m))$ operations, when m is large where $s = |R|$. The conventional direct evaluation of $L_A(\mathbf{x})$ involves mn multiplications and $m(n-1)$ additions, so an improvement is seen when s is small relative to m.

Polynomial evaluation is examined in Section IV and the algorithm for linear forms is combined with a decomposition of a polynomial into a vector-matrix-vector multiplication to show that every polynomial of degree n whose coefficients are taken from a set of s elements can be realized with about $\sqrt{n}\,s$ scalar multiplications, $2\sqrt{n}$ nonscalar multiplications, and $O(n/\log_s(n))$ additions, when n is large. The polynomial decomposition is similar to one used by Paterson and Stockmeyer (Ref. 2), and it achieves about the same number of nonscalar multiplications but uses fewer scalar multiplications and additions.

In Section V, a simple counting argument is developed to show that the upper bounds derived in earlier sections are sharp for matrix-vector multiplications by "chains," that is, straight-line algorithms.

II. The Algorithm

The algorithms for computing $L_A(\mathbf{x})$, where $|R| = s$, will be given in terms of an algorithm $\mathcal{B}$ for the construction of all distinct linear forms in $y_1, y_2, \cdots, y_k$ with coefficients from R. That is, $\mathcal{B}$ computes $L_B(\mathbf{y})$ where B is the $s^k \times k$ matrix with s^k distinct rows and entries from R. The algorithm $\mathcal{A}$ for $L_A(\mathbf{x})$ will use several versions of $\mathcal{B}$.

The algorithm $\mathcal{B}$ has two steps. Let $R = \{\alpha_1, \alpha_2, \cdots, \alpha_s\}$. Then,

Step 1: form $\alpha_i \cdot y_j; \qquad 1 \leq i \leq s, 1 \leq j \leq k.$

Step 2: let $S(i_1, i_2, \cdots, i_\ell) =$

$$\alpha_{i_1} \cdot y_1 + \alpha_{i_2} \cdot y_2 + \cdots + \alpha_{i_\ell} \cdot y_\ell;$$
$$1 \leq \ell \leq k, 1 \leq i_j \leq s.$$

Each element of $L_B(\mathbf{y})$ is equal to $S(i_1, i_2, \cdots, i_k)$ for some set $\{i_1, i_2, \cdots, i_k\}$ of not necessarily distinct integers in $\{1, 2, \cdots, s\}$. Construct $S(i_1, i_2, \cdots, i_\ell)$ recursively from

$$S(i_1) = \alpha_{i_1} \cdot y_1$$

and

$$S(i_1, i_2, \cdots, i_\ell) = S(i_1, i_2, \cdots, i_{\ell-1}) + \alpha_{i_\ell} \cdot y_\ell$$

for $2 \leq \ell \leq k$.

The first step uses $\pi_B = ks$ scalar multiplications. Let $N(s, \ell)$ be the number of additions to construct all linear forms $S(i_1, i_2, \cdots, i_\ell)$. Then, from step 2, it follows that

$$N(s, 1) = 0$$

$$N(s, \ell) = N(s, \ell - 1) + s^\ell$$

From this we conclude that

$$N(s, \ell) = [(s^{\ell+1} - 1)/(s-1)] - (s+1) \leq s^{\ell+1}$$
$$\geq s^\ell$$

for $s \geq 2$, and $\ell \geq 2$.

Therefore, the number σ_B of additions to form $L_B(\mathbf{y})$ satisfies $s^k \leq \sigma_B \leq s^{k+1}$.

Partition A into

$$A = [B_1, B_2, \cdots, B_p]$$

where $B_1, \cdots, B_{p-1}$ are $m \times k$, B_p is $m \times (n - (p-1)k)$, and $p = \lceil n/k \rceil$. Similarly, partition $\mathbf{x} = \mathbf{y}^1, \mathbf{y}^2, \cdots, \mathbf{y}^p$ where $\mathbf{y}^r = (x_{(r-1)k+1}, \cdots, x_{rk})$ for $1 \leq r \leq p-1$ and $\mathbf{y}^p$ is suitably defined. It follows that

$$A\mathbf{x} = B_1\mathbf{y}^1 + B_2\mathbf{y}^2 + \cdots + B_p\mathbf{y}^p \qquad (*)$$

where + denotes column vector addition.

The algorithm $\mathcal{A}$ for $L_A(\mathbf{x})$ has two steps.

Step 1: construct $L_{B_r}(\mathbf{y}^r)$, $1 \leq r \leq p$, using $\mathcal{B}$, that is, identify the linear forms corresponding to rows of B_r and choose the appropriate forms from those generated by $\mathcal{B}$ on $\mathbf{y}^r$.

Step 2: construct $L_A(\mathbf{x})$ by adding as per $(*)$ above.

The number of multiplications used by $\mathcal{A}$ is $\pi_A = ns$. The number of additions used in step 1 is no more than $p\sigma_B$, and in step 2 it is no more than $m(p-1)$. Therefore, the number of additions used by $\mathcal{A}$ satisfies

$$\sigma_A \leqq p\,s^{k+1} + m(p-1)$$

where $p = \lceil n/k \rceil$. Ignoring diophantine constraints and with $k = \log_s(m/\log_s(m))$, we have

Theorem 1. For each $m \times n$ matrix A over a finite set R of cardinality $|R| = s$, the m "linear forms"

$$L_A(\mathbf{x}) = \{a_{i1}\cdot x_1 + \cdots + a_{in}x_n: \qquad 1 \leqq i \leqq m\}$$

can be computed with $\pi_A = ns$ multiplications and $\sigma_A \leqq O(mn/\log_s(m))$ additions when m is large relative to s.

Proof: Ignoring diophantine constraints, we have

$$\sigma_A \leqq n\left(\frac{s^{k+1} + m}{k}\right)$$

and $s^k = m/\log_s(m)$. Therefore,

$$\sigma_A \leqq \frac{nm}{\log_s(m)}(1+\varepsilon_1)/(1-\varepsilon_2)$$

where $\varepsilon_1 = s/\log_s(m)$ and $\varepsilon_2 = \log_s\log_s(m)/\log_s(m)$. If $\varepsilon_2 < ½$, it is easily shown that $(1-\varepsilon_2)^{-1} \leqq 1 + 2\varepsilon_2$. Also,

$$(1+\varepsilon_1)(1+2\varepsilon_2) \leqq 1 + 2(\varepsilon_1 + \varepsilon_2)$$

if $\varepsilon_2 < ½$, which holds for $m \geqq 16$ when $s \geqq 2$. It follows that

$$\sigma_A \leqq \frac{nm}{\log_s(m)}(1 + 2(\varepsilon_1 + \varepsilon_2))$$

when $m \geqq 16$. Since ε_1 and ε_2 approach zero with increasing m, the conclusion of the theorem follows. Q.E.D.

When $m >> s$, this result represents a distinct improvement over the conventional algorithm for evaluating $L_A(\mathbf{x})$, which uses m scalar multiplications and $m(n-1)$ additions. It should be noted that the reduction in the number of additions by a factor of $\log_s(m)$ obtained with algorithm $\mathcal{A}$ follows directly from a reduction by a factor of about k in algorithm $\mathcal{B}$. The obvious algorithms for $L_B(\mathbf{y})$ uses ks^k additions, but $\mathcal{B}$ computes it with no more than s^{k+1} additions.

Although algorithm $\mathcal{A}$ (and $\mathcal{B}$) was discovered independently by the author, it does represent a generalization of an algorithm of Kronrod reported in Arlazarov, et al. (Ref. 3). His result applies to the multiplication of two arbitrary Boolean matrices. The heart of algorithm $\mathcal{A}$ is algorithm $\mathcal{B}$ and this was known to the author (Ref. 4) in the context of the calculation of all Boolean minterms in n variables. This will be discussed in the next section.

III. Applications

In the set $L_A(\mathbf{x})$ of linear forms, the elements a_{ij} and x_j are uninterpreted as are the operations of multiplication and addition. By attaching suitable interpretations, it is seen that algorithm $\mathcal{A}$ for linear forms has applications to many different problems, several of which are now described.

A. Multiplication of a Vector by a Known Matrix

Let R be a set of s variables over S, $R = \{z_1, z_2, \cdots, z_s\}$, and let $S = T = \{\text{reals}\}$. Let + and $\cdot$ be addition and multiplication on the reals. Then $L_A(\mathbf{x})$ represents multiplication of $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ by a known (but not fixed) matrix A. That is,

$$L_A(\mathbf{x}) = \{z_{k_{i1}}\cdot x_1 + z_{k_{i2}}\cdot x_2 + \cdots + z_{k_{in}}\cdot x_n: \qquad 1 \leqq n \leqq m\}$$

where the $m \times n$ matrix of indices $\{k_{ij}\}$ is fixed. For any given matrix $\{k_{ij}\}$, $L_A(\mathbf{x})$ can be computed using ns real multiplications and $O(mn/\log_s(m))$ real additions.

Independent evaluation of the m forms requires a total of at least $m(n-1)$ operations for any s, since each form consists of n functionally independent terms.

Special Cases:

(1) $z_i, z_2, \cdots, z_s$ are assigned distinct real values.

(2) $s = 2$, $z_1 = 0$, $z_2 = 1$. Then, $L_A(\mathbf{x})$ is a set of subset sums such as

$$\{x_1 + x_3, \qquad x_2 + x_3 + x_4\}$$

NOTE: Concerning Case (1), Winograd (Ref. 1) has shown that there exist fixed real (and unrestricted) $m \times n$ matrices and vectors $\mathbf{x}$, such that mn real multiplications and $m(n-1)$ real additions are required for their computation with "straight-line" algorithms. Thus, a significant savings is possible if the matrix entries can assume at most s distinct real values, and s is small relative to m.

B. Matrix-Matrix Multiplication *AX*, *A* Known

Let R and S be as above and let T be the p-fold cartesian product Q^p, $Q = \{\text{reals}\}$. Let $\cdot$ be conventional scalar multiplication (consisting of p real multiplications) and let $+$ be vector addition on the reals (consisting of p real additions). Then, $L_A(\mathbf{x})$ represents multiplication of the $n \times p$ matrix X over the reals by a known (but not fixed) $m \times n$ matrix A. That is,

$$L_A(\mathbf{x}) = \{z_{k_{i1}} \cdot \overline{x}_1 + z_{k_{i2}} \cdot \overline{x}_2 + \cdots + z_{k_{in}} \cdot \overline{x}_n : \quad 1 \leq i \leq m\}$$

where $\overline{x}_\ell$ denotes the ℓ-th row of X and the $m \times n$ matrix of indices $\{k_{ij}\}$ is fixed. For any given matrix $\{k_{ij}\}$, $L_A(\mathbf{x})$ can be computed using nps real multiplications and $O(mnp/\log_s(m))$ real additions.

NOTE: When $n = m = p$, Strassen's algorithm (Ref. 5) for matrix-matrix multiplication can be used at the cost of at most $(4.7)\, n^{\log_2 7}$ binary operations. As a consequence, Strassen's algorithm is asymptotically superior to algorithm $\mathcal{A}$ for this problem. However, when $s = 2$, algorithm $\mathcal{A}$ is the superior algorithm for $n \leq 10^{10}$!! Moral: beware of arguments based upon asymptotics.

C. Boolean Matrix Multiplication

Let $R = S = \{0, 1\}$, $T = \{0, 1\}^p$, let $\cdot$ be Boolean vector conjunction, and let $+$ be Boolean vector disjunction. Then, $L_A(\mathbf{x})$ represents the multiplication of a known Boolean $m \times n$ matrix A by an arbitrary Boolean $n \times p$ matrix X. That is,

$$L_A(\mathbf{x}) = \{a_{i1} \cdot \overline{x}_1 + a_{i2} \cdot \overline{x}_2 + \cdots + a_{in} \cdot \overline{x}_n : \quad 1 \leq i \leq m\}$$

where $\overline{x}_\ell$ is the ℓ-th row of the $n \times p$ matrix X. The algorithm for computing AX uses no multiplications and $O(mnp/\log_2(m))$ additions.

NOTE: If A is an arbitrary Boolean matrix and if the selection procedure of step 1 of algorithm $\mathcal{B}$ can be executed without cost, the algorithm of Kronrod (Ref. 3) results. The number of operations performed, exclusive of selection, equals that given above. The Kronrod algorithm uses more operations because of a poor choice of the parameter k.

D. Boolean Minterms

Let $R = S = T = \{0, 1\}$, let $+$ be the Boolean conjunction, and let $\cdot$ be defined by

$$\alpha \cdot x = \begin{cases} x & \alpha = 1 \\ \overline{x} & \alpha = 0 \end{cases}$$

where $\overline{x}$ denotes the Boolean inverse. Then, $L_A(\mathbf{x})$ represents a set of minterms such as

$$\{\overline{x}_1 x_2 x_3,\ \overline{x}_1 x_2 \overline{x}_3,\ x_1 \overline{x}_2 \overline{x}_3\}$$

NOTE: The set of all 2^n distinct minterms, suitably ordered, represents a map from the binary to positional representation of the integers $\{0, 1, 2, \cdots, 2^n - 1\}$. This map can be realized with at most 2^{n+1} conjunctions and is a map that is useful in many constructions, such as those in Ref. 4.

E. Products in a Group G

Let $R = \{-1, 0, 1\}$, $S = T = G$, let $\alpha \cdot x = x^\alpha$ (raise to a power), and let $+$ be group multiplication. Then $L_A(\mathbf{x})$ represents a set of m products of n terms each. For example,

$$\{a b^{-1} c d^{-1}, b c^{-1}, a^{-1} c^{-1} d^{-1}\}$$

is such a set, where x^{-1} is the group inverse of x and x^0 is the group identity that is suppressed.

IV. Polynomial Evaluation

We turn next to the evaluation of polynomials of degree n. Let

$$p(x) = a_0 + a_1 \cdot x^1 + a_2 \cdot x^2 + \cdots + a_n x^n$$

where $+$ and $\cdot$ represent vector addition and scalar multiplication, where $x^1 = x$, $x^i = x * x^{i-1}$, and $*$ represents vector multiplication. Let $a_i \varepsilon R$, $x \varepsilon T$ and

$$\begin{aligned} \cdot &: R \times T \to T \\ + &: T \times T \to T \\ * &: T \times T \to T \end{aligned}$$

where $+$ and $*$ are associative and $*$ distributes over $+$. We shall construct an algorithm $\mathcal{P}$ for polynomial evaluation, which employs algorithm $\mathcal{A}$ for linear forms.

Algorithm $\mathcal{P}$ has three steps. Without great loss of generality, let $n = k\ell - 1$, and assume that $\mathbf{a} = (a_0, a_1, \cdots, a_n)$ has entries from a set of size s.

Step 1: construct $x^2, x^3, \cdots, x^\ell$.

Step 2: construct the k linear forms in $1, x, x^2, \cdots, x^{\ell-1}$

$$\begin{bmatrix} r_0(x) \\ r_1(x) \\ \cdot \\ \cdot \\ \cdot \\ r_{k-1}(x) \end{bmatrix} = \begin{bmatrix} a_0 a_1 & \cdots & a_{\ell-1} \\ a_\ell a_{\ell+1} & \cdots & a_{2\ell-1} \\ & & \\ & & \\ & & \\ a_{(k-1)\ell} & \cdots & a_{k\ell-1} \end{bmatrix} \begin{bmatrix} 1 \\ x^2 \\ \cdot \\ \cdot \\ \cdot \\ x^{\ell-1} \end{bmatrix}$$

using algorithm $\mathcal{A}$.

Step 3: construct $p(x) = r_0(x) + r_1(x) * x^\ell + \cdots + r_{k-1}(x) * x^{(k-1)\ell}$ using Horner's rule. Let σ_p denote the number of vector additions used by $\mathcal{P}$, π_p the number of scalar multiplications and μ_p the number of vector multiplications. Since the forms required in step 1 can be realized with $\ell - 1$ vector multiplications and step 3 with $k - 1$ such multiplications and $k - 1$ additions, we have

$$\sigma_p \leq O((n+1)/\log_s(k)) + k - 1$$

$$\pi_p = \ell s$$

$$\mu_p = \ell + k - 2$$

since $k\ell = n + 1$. If we ignore diophantine constraints and choose $k = \sqrt{(n+1)}$, to minimize μ_p we have:

Theorem 2. For each $\mathbf{a} = (a_0, a_1, \cdots, a_n) \varepsilon R^{n+1}$ with R a set of cardinality $|R| = s$, $p(x) = a_0 + a_1 x + \cdots + a_n x^n$ can be evaluated with σ_p vector additions, π_p scalar multiplications and μ_p vector multiplications where

$$\sigma_p \leq O(n/\log_s(n))$$

$$\pi_p \leq s\sqrt{n+1}$$

$$\mu_p \leq 2\sqrt{n+1} - 2$$

when n is large relative to s.

Horner's rule, which is the optimal procedure for evaluating an arbitrary polynomial on the reals, uses n multiplications and n additions. Even when $\mathbf{a}$ and x assume fixed real values, there exist vectors $\mathbf{a}$ and values x for which Horner's rule is still optimal (Ref. 1). When the coefficients are drawn from a set of size s, however, Horner's rule can be improved upon by a significant factor when n is large relative to s.

The decomposition of $p(x)$ used by algorithm $\mathcal{P}$ is very similar to that used by Paterson and Stockmeyer (Ref. 2) in their study of polynomials with rational coefficients. They have shown that $O(\sqrt{n})$ vector multiplications are necessary and sufficient for such polynomials, but their algorithms use $O(n)$ additions. Algorithm $\mathcal{P}$ achieves $O(\sqrt{n})$ vector multiplications but requires only $O(n/\log_s(n))$ additions when n is large relative to s. Clearly, algorithm $\mathcal{P}$ can be applied to any problem involving polynomial forms.

V. Some Lower Bounds

The purpose of this section is to demonstrate the existence of problems for which the performance of algorithm $\mathcal{A}$ can be improved upon by at most a constant factor. To do this, we must carefully define the class of algorithms that are permissible. Then, we count the number of algorithms using C or fewer operations and show that if C is not sufficiently large, not all problems of a given type (such as matrix-vector multiplication) can be realized with C or fewer operations.

A *chain* β is a sequence of steps $\beta_1, \beta_2, \cdots, \beta_L$ of two types: *data steps*, in which $\beta_i \varepsilon \{y_1, y_2, \cdots, y_n\} \cup K (y_i \notin K, y_i \neq y_j, i \neq j$ and $K \subset Q$ is a finite set of constants), or *computation steps*, in which

$$\beta_i = \beta_j \circ \beta_k \qquad j, k < i$$

and $\circ : Q \times Q \rightarrow Q$ denotes an operation in a set Ω.

Associated with each step β_i of a chain is a function $\bar{\beta}_i$, which is β_i if β_i is a data step and

$$\bar{\beta}_i = \circ(\bar{\beta}_j, \bar{\beta}_k)$$

if β_i is a computation step. Clearly, $\bar{\beta}_i \cdot Q^n \rightarrow Q$. A chain β is said to *compute* m functions $f_1, f_2, \cdots, f_m, f_i: Q^n \rightarrow Q$ if there exists a set of m steps $\beta_{i_1}, \cdots, \beta_{i_m}$ such that $\bar{\beta}_{i_\ell} = f_\ell, 1 \leq \ell \leq m$.

We now derive an upper bound on the number $N(C, m, n)$ of sets of m functions $\{f_1, \cdots, f_m\}$ that can be realized by chains with C or fewer computation steps.

Lemma. $N(C, m, n \leq v^{4v}, v = C + n + m + |K| + 1$ if $C \geq |\Omega| \geq 2$.

Proof: A chain will have $1 \leq d \leq n + |K|$ data steps and without loss of generality they may be chosen to precede computation steps. Similarly, the order of their appearance is immaterial, so there are at most

$$\binom{n + |K|}{d} \leq 2^{n+|K|}$$

ways to arrange the d data steps.

Let the chain have t computation steps. Each step may correspond to at most $|\Omega|$ operations and each of the two operands may be one of at most $t + d$ steps. Thus, there are at most $|\Omega|^t (t + d)^{2t}$ ways to assign computation steps and at most $2^{n+|K|} |\Omega|^t (t + d)^{2t}$ chains with d data steps and t computation steps. A set of m functions can be assigned in at most $(t + d)^m$ ways.

Combining these results, we see that the number of distinct sets of m functions that can be associated with chains that have C or fewer computation steps is at most

$$\begin{aligned} N(C,m,n) &\leq \sum_{d=1}^{n+|K|} \sum_{t=1}^{C} 2^{n+|K|} |\Omega|^t (t + d)^{2t+m} \\ &\leq (n + |K|)\, C\, 2^{n+|K|} |\Omega|^C (C + n + |K|)^{2C+m} \end{aligned}$$

But

$$(n + |K|)\, 2^{n+|K|} \leq (C + n + |K|)^{C+n+|K|}$$

if $C \geq 2$, and

$$|\Omega|^C \leq (C + n + |K|)^C$$

if $C \geq |\Omega|$, from which it follows that

$$\begin{aligned} N(C,m,n) &\leq (C + n + |K|)^{4C+n+|K|+m+1} \\ &\leq v^{4v} \end{aligned}$$

where $v = C + n + m + |K| + 1$. Q.E.D.

In the interest of deriving a bound quickly, the counting arguments given above are loose. Nevertheless, the bound can at best be improved to about v^v. As seen below, this means a factor of about 4 loss in the complexity bound.

Consider the computation of m subset sums of $\{x_1, x_2, \cdots, x_n\}$, $x_i \,\varepsilon\, \{\text{reals}\}$, as defined in Special Case (2) of Subsection III-A. In the chain defined above let $Q = \{\text{reals}\}$ and let $\Omega = \{+, \text{addition on the reals}\}$. There are 2^n distinct subset sums and the number of sets of m distinct subset sums is the binomial coefficient

$$F = \binom{2^n}{m}$$

Fix $0 < \varepsilon < 1$. If C, n, and m are such that $N(C, n, m) \leq F^{1-\varepsilon}$, then there exists at least one set of m distinct subset sums that require C or more additions.

Theorem 3. Algorithm $\mathscr{A}$ is sharp for some problems, that is, there exist problems, namely, the computation of m subset sums over the reals, that require $O(mn/\log_2(m))$ operations with any chain or "straight-line" algorithm, when $m = O(n)$.

Proof: Set $v^{4v} = F^{1-\varepsilon}$ where $v = C + n + m + |K| + 1$. Then, $N(C, m, n) \leq F^{1-\varepsilon}$. For large F, the solution for v is

$$v = \left(\frac{1}{4} \ln F^{1-\varepsilon}\right) \Big/ \ln\left(\frac{1}{4} F^{1-\varepsilon}\right)$$

Since $m = O(n)$, it can be shown from Stirling's approximation to factorials and an examination of the binomial coefficient F that $\ln F$ is asymptotic to $nm \ln(s)$. From this the conclusion follows. Q.E.D.

The counting argument given above could also be applied to matrix-vector multiplication, Subsection III-A, and to polynomial evaluation on the reals, as described in Section IV, to show that the upper bounds given for these problems are also sharp.

VI. Conclusions

The algorithm presented here for the evaluation of a set of linear forms derives its importance from the minimal set of conditions required of the two operations. In fact, the only condition required is that addition be associative. As a consequence, the algorithm applies to a large class of apparently disparate problems having in common the fact that they can be represented in terms of linear forms of this general nature.

The algorithm allows us to treat two important problems, matrix multiplication with a known matrix and polynomial evaluation with a known polynomial. In both cases an algorithm is constructed that depends explicitly on the matrix entries and the polynomial coefficients. When the entry set of the matrix or of the polynomial coefficients is fixed and the dimensions of either problem are large, a sizable savings in the number of required computations is obtained.

The generality of the algorithm for evaluation of linear forms suggests that it may have application to many important problems not mentioned in this paper.

Acknowledgment

The author acknowledges several important conversations with Dr. Charles M. Fiduccia, which resulted in the generalization and clarification of the principal algorithm for linear forms.

References

1. Winograd, S., "On the Number of Multiplications Necessary to Compute Certain Functions," *Comm. Pure and Applied Math.*, Vol. 23, pp. 165–179, 1970.
2. Paterson, M., and Stockmeyer, L., "Bounds on the Evaluation Time for Rational Polynomials," in *Proceedings of the Twelfth IEEE Symposium on Switching and Automata Theory*, pp. 140–143, Oct. 1971.
3. Arlazarov, V. L., et al., "On Economical Construction of the Transitive Closure of an Oriented Graph," *Soviet Mathematics Doklady*, Vol. 11, No. 5, pp. 1209–1210, 1970.
4. Savage, J. E., "Computational Work and Time on Finite Machines," *J. Assoc. Comput. Mach.*, Vol. 19, No. 4, p. 673, Oct. 1972.
5. Strassen, V., "Gaussian Elimination is Not Optimal," *Numer. Math.*, Vol. 13, pp. 354–356, 1969.